COL774 ASSIGNMENT 1

Aryan Jain
2019CS10334

Q1. Linear Regression

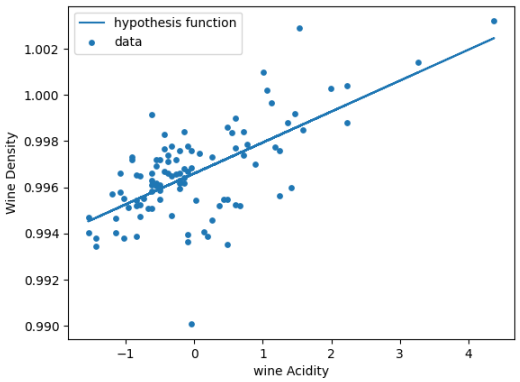
* We implement least squares linear regression to predict wine density based on its acidity.
* The error metric used is :

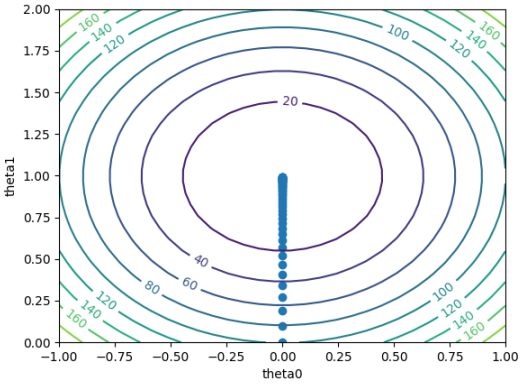
$$J(\Theta) = \frac{1}{2m} * \sum_{i=1}^{m} (y^{(i)} - h_{\Theta}(x^{(i)}))^2$$

* The stopping criteria used is that when the improvement in the error function (decrement in $J(\Theta)$) is less than 1e-8, we consider the function to have converged.

* In each of the graphs of the movement of theta along the contours of $J(\Theta)$, we have plotted the first 50 iterations for visibility and easy comparability.

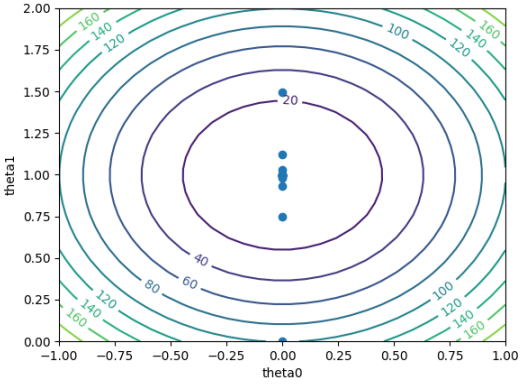
* After trying different values for the learning rate and stopping criteria, the best results were obtained for a learning rate of 0.1 and stopping criteria of 1e-8.
The final mean squared error is:
$J(\Theta)$ = 0.00023899
The final parameters are:
[0.00134017 , 0.99660080]





* For a learning rate of greater than or equal to 2 the function diverges.

* For a learning rate of less than 2 but greater than 1 like 1.1 or 1.5, the function oscillates around the minima before converging.
For a learning rate of 1.5
The final mean squared error is:
$J(\Theta)$ = 0.00023896
The final parameters are:
[0.00134019 , 0.99661629]

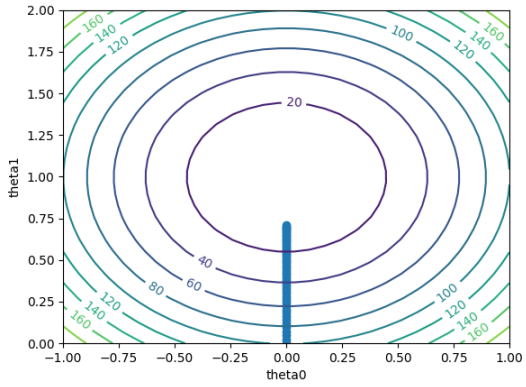


* For a learning rate of 0.025, the function converges in a straightforward manner but the learning rate is much slower with not much gain in accuracy of predictions.
The final mean squared error is:

$J(\Theta)$ = 0.00023914

The final parameters are:

[0.00134014 , 0.99657711]

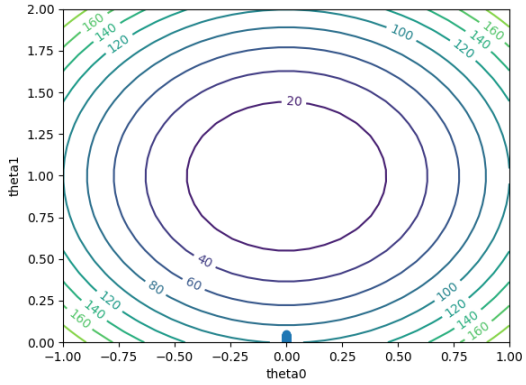


* Similar to a learning rate of 0.025, when we use a learning rate of 0.001, the function takes extremely long to converge which can be seen by the minimal movement in theta in the plot below.
The final mean squared error is:

$J(\Theta)$ = 0.00024394

The final parameters are:

[0.00134013 , 0.99657711]



* The animations are all available in the plots folder.

Q2. Stochastic Gradient Descent

* The learning rate is fixed to 0.01 in this question. We rather use different batch sizes and compare the rate at which convergence is achieved and the accuracy of the achieved hypothesis.
* The data set has 1000000 data points sampled such that :

theta = [[3] , [1] , [2]]

x1 ~ N(3, 4), x2 ~ N(-1, 4), x0 = 1

noise ~ N(0, $\sqrt{2}$)

y = theta.x + noise

* For each batch size, the convergence criteria used is as below:

We consider the function to have converged when the difference in the average squared error per data_point for the last 1000 batches (or for all batches from this epoch, whichever is smaller) of this and the last epoch is less than the stopping_criteria value.

*For each batch size, we observe the:
1. Time to converge
2. Difference in test error of original and trained hypothesis
3. Plot showing movement of theta
4. Number of iterations of the entire dataset required (epochs).
5. Final parameters

* For a batch size of 1000000, that is each epoch has only one batch

Time to converge =

extremely long (> 5 minutes)

Test error of original hypothesis =

0.009829469214999999

Test error of trained hypothesis =

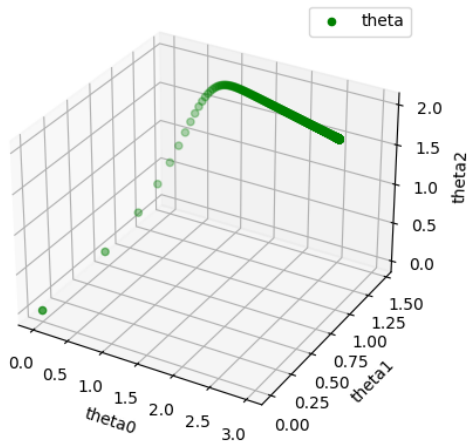
0.009868230105150854

Difference of test errors =

3.87608901508546e-05

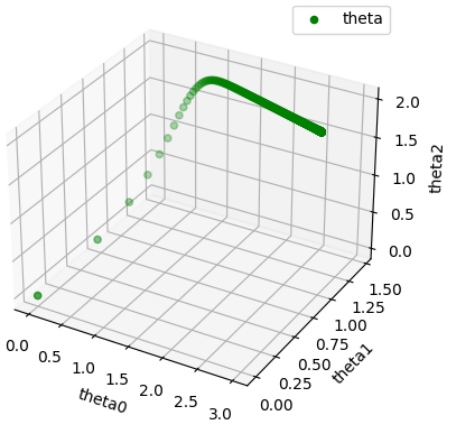
Number of epochs = 16011

Final parameters =
[[2.9658903], [1.00847416], [1.99842191]]



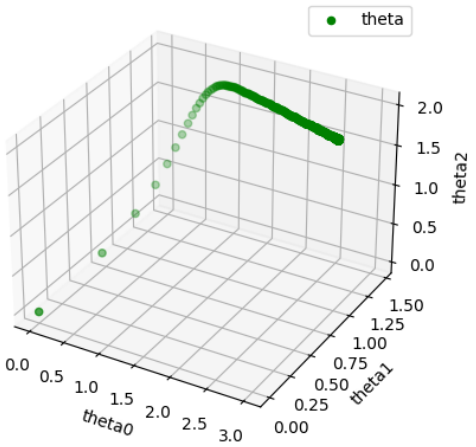
Since each batch contains the entire data set, the model converges very slowly as theta is updated only once per epoch. However, since each update to theta considers the entire data set, convergence is achieved in a very straightforward manner with no movement of theta in unwanted manners.

* For a batch size of 10000, that is each epoch has 100 batches
Time to converge =
Fast (15 seconds)
Test error of original hypothesis =
0.009829469214999999
Test error of trained hypothesis =
0.009831724217716298
Difference of test errors =
2.255002716299026e-06
Number of epochs = 247
Final parameters =
[[2.99431478], [1.00138401], [1.99905271]]



Each epoch contains 100 batches and each batch is much smaller than a batch size of 1e6, theta is updated much faster (100 times per epoch). This leads to convergence in much less time and epochs. The difference is error is much lesser and the trained theta values are much closer to the original theta values for the batch of size 10000 compared to the batch of size 1 million.
However, the convergence does show slightly more variation since each update considers only a part (although significantly large part) of the data set.

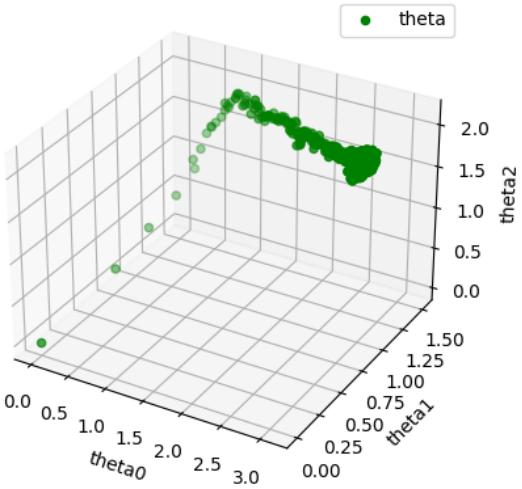
* For a batch size of 100, that is each epoch has 10000 batches, and only the last 1000 of each epoch are considered when calculating convergence
Time to converge =
Extremely Fast (<2 seconds)
Test error of original hypothesis =
0.009829469214999999
Test error of trained hypothesis =
0.00984126401394585
Difference of test errors =
1.1794798945851184e-05
Number of epochs = 5
Final parameters =
[[2.99982912], [0.9995302], [1.99642784]]



The batch size is much smaller which leads to much more updates to theta per epoch and hence much faster convergence.
The smaller batch size does cause a loss in accuracy in comparison to a batch size of 10000, and the movement of theta is also much more random.

* For a batch size of 1, that is each epoch has 1 million batches, and only the last 1000 of each epoch are considered when calculating convergence

Time to converge =
Extremely Fast (<2 seconds)
Test error of original hypothesis =
0.009829469214999999
Test error of trained hypothesis =
0.00984126401394585
Difference of test errors =
1.1794798945851184e-05
Number of epochs = 5
Final parameters =
[[2.99982912], [0.9995302], [1.99642784]]



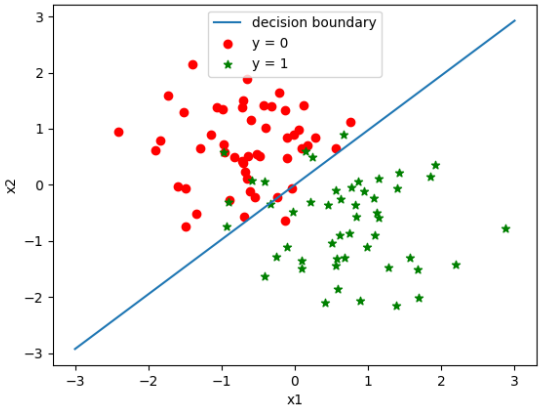
Q3. Logistic Regression

* The error function used is the negative log likelihood function represented as:

$$L(\Theta) = \sum_{i=1}^{m} \{y^{(i)} * log(h_\Theta(x^{(i)})) + (1 - y^{(i)}) * log(1 - h_\Theta(x^{(i)}))\}$$

* We use Newton's method to optimize the negative log likelihood function of the logistic regression. We only need to update theta once to achieve the required parameters.

* The final parameters obtained are:
[[-3.34899391e-18], [8.97341408e-03], [-9.20112283e-03]]



Q4. Gaussian Discriminant Analysis

* We perform gaussian discriminant analysis on the given data. Here the features x0 and x1 are the growth ring diameters of a salmon in freshwater and marine water, whereas the classification separates the fishes into 2 categories, Alaska and Canada. For our analysis, we will represent Alaska as class 0 and Canada as class 1.

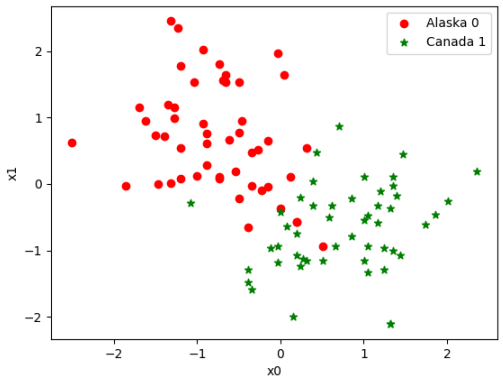
* The means as calculated are:
μ_0= [[-0.75529433 , 0.68509431]]
μ_1= [[0.75529433 , -0.68509431]]

* If the covariances are considered to be equal then the overall covariance is:
Σ = [[0.42953048 -0.02247228]
[-0.02247228 0.53064579]]

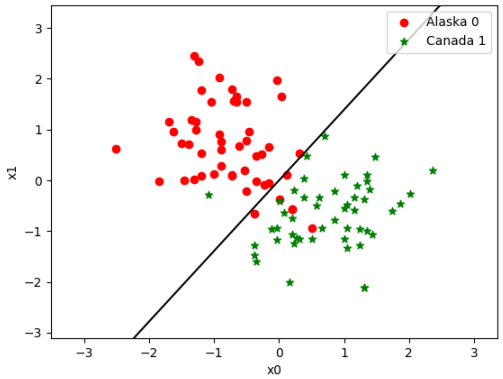
* The actual covariances are:
Σ0 = [[0.38158978 -0.15486516]
[-0.15486516 0.64773717]]
Σ1 = [[0.47747117 0.1099206]
[0.1099206 0.41355441]]

* Phi is : ϕ = 0.5

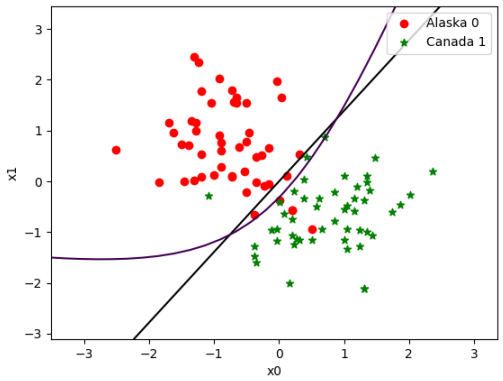
* Plot of the training data



* Plot of the linear decision boundary found considering equal covariances



* Plot of the quadratic decision boundary learnt by considering actual covariances



* The linear boundary only considers the means of the two classes and does not take into account the covariances, which contains the information about how densely packed the data points of a certain class are. This information is considered by the quadratic decision boundary, hence it can classify the data much more accurately and confidently compared to the linear decision boundary.
From the graph above, we can see that the quadratic boundary misclassified much fewer data points compared to the linear decision boundary.
And when we look at the probability values as well, the linear model has much lower values compared to the quadratic model. Hence the quadratic model classifies the data much more confidently and with clearer boundaries.